Running Head: VOLUNTEER COMPUTING GAMES

Volunteer Computing Games: Merging Online Casual Gaming with Volunteer Computing

Charles A. Cusack, Hope College
Evan Peck, Tufts University
Maria Riolo, California Institute of Technology

Abstract

We describe *volunteer computing game*s, a new paradigm for computing which merges volunteer computing and online casual games. In volunteer computing, the main goal is to harness the computational power of many users' computers to solve a large computational problem. Unfortunately, participants in volunteer computing come from a very limited demographic—primarily middle-aged male computer experts. Since casual games have an almost universal appeal and, consequentially, a broad player base, we argue that volunteer computing games will significantly impact volunteer computing efforts by attracting and retaining more users.

We describe one framework for implementing volunteer computing games to solve problems whose solution space can be modeled as a search tree. Using our prototype game as an example, we discuss general design principles that are of particular importance with volunteer computing games. We argue that the full power of volunteer computing games occurs when human computation is added to the mix. An ideal volunteer computing game will attract men and women of all ages and harness the computing power of both the users and their computers. We conclude with a very brief discussion of volunteer computing within Massively Multiplayer Online Games (MMOGs).

Volunteer Computing Games: Merging Online Casual Gaming with Volunteer Computing

Introduction

As scientists have become more frequently confronted by increasingly complex problems, research teams have turned to computers to aid in the exploration of these challenges. Unfortunately, technology has simply not kept pace. A single processor is no longer adequate for the problems researchers now encounter. As a result, a research team may not receive tangible results for weeks, months, or even years.

One solution is supercomputing. However, this is not a practical choice for many researchers. Although supercomputers provide improved computing power, they are both uncommon and expensive. Thus, recent efforts have started to look towards parallel processing and distributed computing. A second option is the use of a computing cluster. But clusters are also not without their limitations. Since clusters contain a predetermined number of nodes, the system's processing power has an inherent ceiling.

A third option is volunteer computing. Volunteer computing is the process that allows people from across the globe to donate their computer resources in a joint effort, effectively creating an extremely powerful supercomputer. Currently, the world's largest contributor to volunteer computing is the Berkeley Open Infrastructure for Network Computing (BOINC). BOINC uses approximately 400,000 volunteer computers, contributing an average of over 400 TeraFLOPS of combined processing power (Anderson & McLeod, 2007). Among the forty projects that BOINC sustains, two of the most popular are Rosetta@home and SETI@home. While Rosetta@home helps determine the 3-dimensional shapes of proteins in the hopes of curing diseases, SETI@home processes digital signals from a radio telescope at the Arecibo radio observatory (Anderson, 2004).

In order to understand the full potential of volunteer computing, consider that as of 2004, approximately 150 million personal computers were connected to the internet—a number that is estimated to exceed one billion by 2015 (Anderson, 2003). Of these machines, consider further that many are idle for nearly ninety percent of each day. Even active computer users typically use less than ten percent of their machine's CPU (Justiniano, 2005). Accordingly, it is only logical for the research community to take advantage of this enormous source of unused computational potential.

Volunteer computing does have its fair share of obstacles, particularly in regards to participation. Theoretically, the number of volunteer computing participants should be limited only by the number of computers online. However, the 400,000 active BOINC users currently compose less than 0.3 percent of the total number of computers connected to the internet. Therefore, taking into account that the average broadband internet user spends 48 percent of their leisure time online (Marketing Charts, 2007), we are exploring combining the accessibility and entertainment of online casual games with the effectiveness of volunteer computing. Casual games are "games that generally involve less complicated game controls and overall complexity in terms of gameplay or investment required to get through the game" (Wallace & Robbins, 2006). The IGDA reports that "the core and casual downloadable games sector grew over 100 percent from 2003-2004" (Wallace & Robbins, 2006). In fact, Nickelodeon recently announced that it will invest 100 million dollars in casual games over the next few years (Androvich, 2007). As casual gaming has become an increasingly popular leisure activity, the innovative possibilities to use them have similarly increased.

Luis von Ahn's research group at Carnegie Mellon is already taking advantage of casual gaming to assist in various human computing tasks, including labeling images on the web with the *ESP Game* (von Ahn & Dabbish, 2004) and *Phetch* (von Ahn, Ginosar, & Kedia, 2006; vonAhn, Ginosar, Kedia, Liu, & Blum, 2007), locating objects in images with *PeekABoom* (von Ahn, Liu, & Blum, 2006), collecting common sense facts with *Verbosity* (von Ahn, Kedia, & Blum, 2006), and annotating sounds and music with *Tag a Tune* (Law, von Ahn, Dannenberg & Crawford, 2007). Recently, Von Ahn introduced several of his games on GWAP (http://gwap,com), a site devoted to what he calls "games with a purpose." As von Ahn (2006) puts it, "Each year, people around the world spend billions of hours playing computer games. What if all this time and energy could be channeled into useful work?" It is important to note that von Ahn is interested in the computational power of the user, not the computer: "Rather than designing a computer vision algorithm that generates natural language descriptions for arbitrary images (a feat still far from attainable), we opt for harnessing humans" (von Ahn, Ginosar, & Kedia, 2006). Thus, von Ahn's games work as *alternatives* to computer algorithms (Thompson, 2007).

Currently, researchers at the University of Washington are developing a game called *Fold It!* (http://fold.it) that allows players to assist in predicting protein structures, an important area of biochemistry that seeks to find cures for diseases. The game presents visual representations of proteins to the player and allows them to use several key algorithms from the Rosetta@home code to find the optimal folding structure of the protein. By tracking the strategies of the best players, they hope to develop new heuristics to be incorporated into Rosetta@home as well as "go well beyond established limitations of all purely computational methods…eventually we want to leverage the human ability to think abstractly, and visually compose 3d structures… effectively tapping into a problem solving part of the brain instead of visual recognition part of the brain." (Z. Popović, personal communication, February 14, 2008). As with von Ahn's research, the idea here is to harness the insights of the players, not necessarily their computers. They believe that by presenting geometrical optimization problems to many users, they can solve them better than by only employing traditional algorithms, since the insights of a few users may provide a heuristic that was previously unknown, or a clever user might stumble upon a better solution than what is already known.

As far as we can tell, there are not similar efforts to utilize the popularity of online casual gaming to assist in non-human computation. Although the *Fold It!* Team may use their game to encourage participation in Rosetta@home, it appears von Ahn's games are not being used to encourage participation in volunteer computing. Thus, we present *Wildfire Wally,* a prototype of a *volunteer computing game*—a casual game which is an implementation of a volunteer algorithm.

In the game, players play as Wally, a red-bearded wilderness personality who is desperately trying to protect his forest from a raging fire. Wally extinguishes the flames by either dousing blazing trees with water, or by creating fire-lines that isolate a burning area of the forest. The player simply clicks on a cell to perform the appropriate action. If the cell contains fire, the fire is partially quenched. If the cell does not contain fire, some of the trees are removed to begin creating a fire line. As players progress from level to level, wind gusts and dropping humidity make containing the fire increasingly difficult. If a certain number of trees are not conserved, the player loses. Each move in *Wildfire Wally* corresponds to a decision in a traversal of a search tree for an instance of the maximum clique problem. This allows each player to exhaust a portion of a potentially enormous search tree. In addition, players have the option of running computation in the background while they play. Thus, many players from around the world can contribute to the solution of the same problem while playing a game.

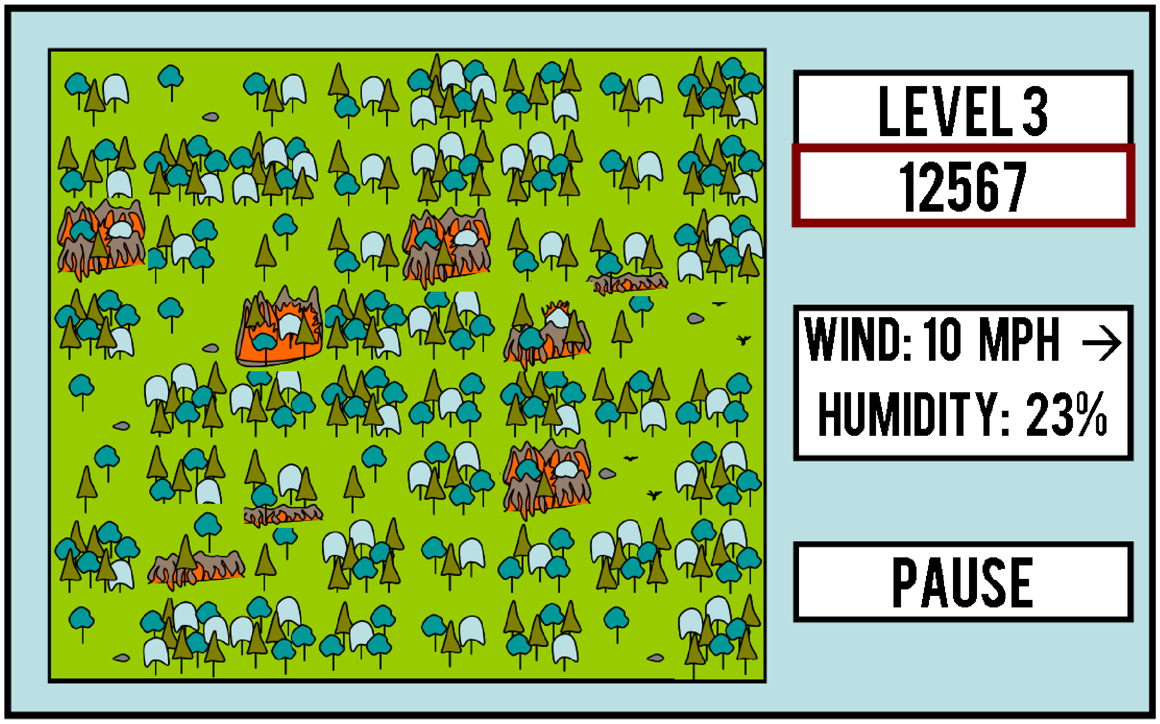

FIGURE 1 A mock-up of *Wildfire Wally*

When properly designed and implemented, a volunteer computing game allows a researcher to harness the computational power of both the player and her computer, as well as introduce her to the idea of volunteer computing in general, and a specific research project. We believe all of these outcomes have the potential to change the face of volunteer computing, both by increasing participation in these projects significantly—translating into added computational power—and increasing the sophistication of the contributions by clever players.

Motivation

Volunteer Computing Limitations

While volunteer computing has proven to be useful to the research community, we have previously expressed that it has fallen far short of its full potential. Less than 0.3 percent of computers participate in volunteer computing at all. Since the success of volunteer computing is entirely dependent on participation, it is a problem that deserves special attention. Currently, there are at least four distinct barriers to participation in volunteer computing.

First, many people have never heard of volunteer computing and do not know that they can contribute to one of over one hundred active projects. This is particularly disappointing considering that the projects are varied enough to appeal to any given individual. If people are unaware of their existence, they cannot participate, no matter how eager they might be.

Second, there is very little motivation for people to participate. Most forms of volunteer computing consist of merely running an application in the background. While individuals can be ranked by their contributions, these contributions correspond to the amount of time running the application rather than any meaningful actions completed by the users. Moreover, these rankings are dominated by teams and individuals with far superior computing resources, making the contributions of the average participant seem insignificant by comparison.

Third, those who are currently active in volunteer computing come from a very limited demographic group. A survey of potential BOINC users indicates that 93.9 percent of 33,988 respondents are male. Only 4.7 percent of respondents fall in the age range of 0-19 years old (BOINC, 2007)

Fourth, many people do not have the technological aptitude to participate. Contributing to volunteer computing often consists of downloading and properly configuring an application. For many users, this either exceeds their limited abilities, or is simply an extra step they are reluctant to take. In fact, about 38 percent of those participating in the survey indicated they do not run BOINC for technical reasons. Furthermore, the same BOINC survey shows that only 3.2 percent of all participants consider themselves “beginners” when it comes to using a computer.

A whopping 56.6 percent consider themselves computer experts (BOINC, 2007). Clearly, there is a divide.

The average participant of BOINC projects is a middle-aged male computer expert. After considering that an estimated one billion people have access to the internet, most of whom are not middle aged male computer experts, it is apparent that the power of volunteer computing has gone largely untapped (Internet World Stats, 2007).

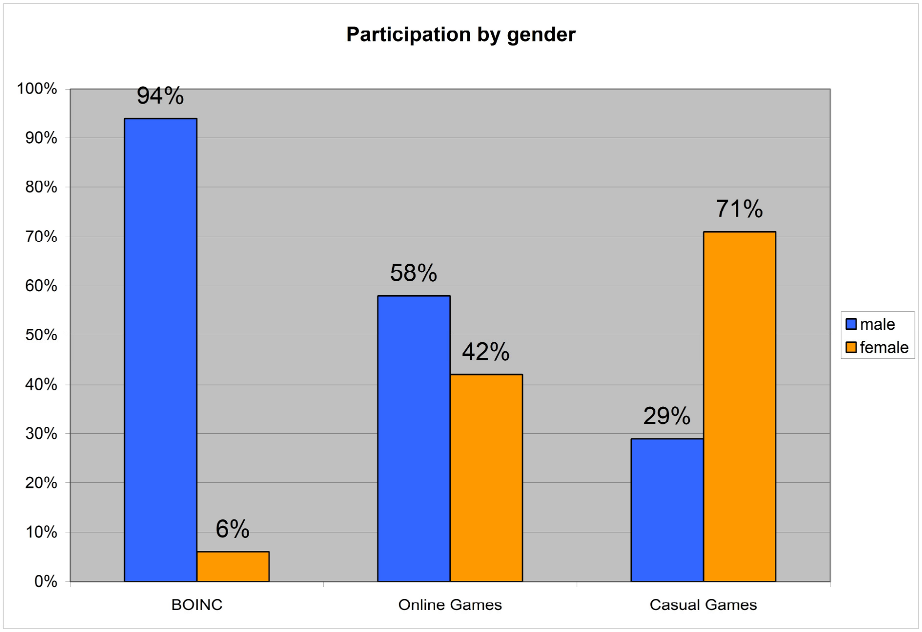

FIGURE 2 Participation by gender in volunteer computing using BOINC (BOINC, 2007), online gaming (ESA, 2006), and casual gaming at trygames.com (Macrovision, 2006).

Benefits of Casual Gaming

While volunteer computing has fallen short of its potential, online casual gaming has flourished. An estimated 100 million people in the United States alone played a computer game in 2006 (Wallace & Robbins, 2006). In addition, the people who participate in casual gaming provide a refreshing divergence from the lopsided demographic involved with volunteer computing. As can be seen in Figure 2, although only 6 percent of volunteer computing participants are female, 42 percent of online gamers are female. This number jumps to 71 percent when we consider only online casual games. In fact, the average casual gamer is a woman in her forties (Wallace & Robbins, 2006). Since different games may appeal to different demographics, it is possible for games to be designed for a specific age group or gender.

Casual gaming is capable of breaking all of the barriers to volunteer computing. It has already been suggested that casual gaming can solve demographic problems, but the steadily growing popularity of casual gaming can also overcome awareness obstacles. This popularity has resulted in the emergence of numerous casual gaming websites, including AddictingGames, MiniClip, Yahoo! Games, and many more. Millions of internet users play games, and we expect that a well designed volunteer computing game would be at least as popular as other casual games. In addition, the fact that the game is contributing to important research might become an added attraction.

The popularity of casual gaming also suggests that very little technical savvy is needed to participate. Many online casual games are played in a web browser and do not generally require users to install any additional software. In addition, casual games have simple rules and control mechanisms—often only requiring the use of the mouse, and maybe a few keys on the keyboard.

Finally, because good games are engaging and entertaining, there is an immediate incentive to participate. In addition to the goals in the game, internet users have the pleasure of knowing that their gameplay contributes towards a larger problem. Participating in volunteer computing becomes enjoyable because it is a pleasant side-effect and not the main focus of playing the game.

A Volunteer Computing Algorithm

Here we present a computational problem—the Maximum Clique Problem—and describe a volunteer computing algorithm capable of solving this problem. We also discuss how the algorithm can be easily adapted to solve any problem that can be modeled using a search tree. In the next section we will discuss how a game can be used to run this algorithm.

The Maximum Clique Problem

Before we can discuss our volunteer computing game, we need to introduce the computational problem to be solved. We chose the maximum clique problem since it is a well known computationally complex problem. We begin with some basic terminology.

A *graph* **G** = (**V**, **E**) is composed of a set of *vertices*, **V**, and a set of pairs of vertices, **E**, called *edges* (see Figure 3a), where |**V**| = *n*. We will label the vertices of a graph 1, 2, …, n. Two vertices of a graph are *adjacent* to each other if they are connected by an edge.

A *complete graph* is a graph in which every vertex is adjacent to every other vertex. A *subgraph* **G'** = (**V'**, **E'**) of **G** = (**V**, **E**) is a graph such that **V'** $\subseteq$ **V** and **E'** $\subseteq$ **E**. A *clique* is a subgraph of a graph that is a complete graph. A *maximum clique* is a clique of largest size in **G**. The *maximum clique problem* is the problem of finding a largest clique in a graph.

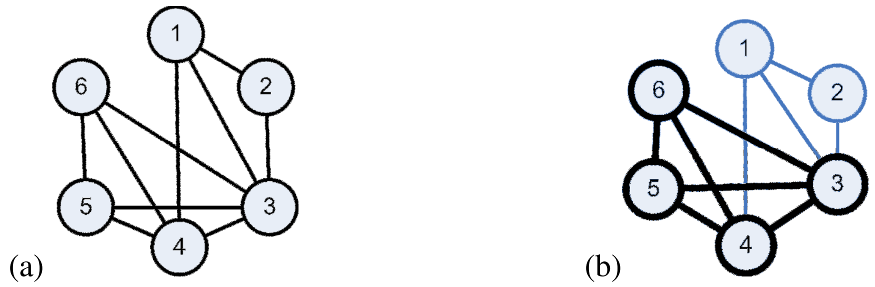

FIGURE 3 (a) A graph with vertex set {1, 2, 3, 4, 5, 6} and edge set {(1,2), (1,3), (1,4), (2,3), (3,4), (3,5), (3,6), (4,5), (4,6), (5,6)}. (b) The same graph with the maximum clique {3, 4, 5, 6} marked in bold.

The Origins of the Volunteer Algorithm

The algorithm we use to solve the maximum clique problem is based on a parallel algorithm by Thimm, Kreher, and Merkey (2006). In this section, we will briefly describe the idea behind the serial algorithm to solve the problem, and in the following section we will describe the volunteer computing algorithm. See (Cusack, 2006) for a more detailed description of the algorithm.

When examining the cliques of a graph, redundancy can vastly reduce efficiency. To illustrate this, consider a clique containing vertices 1, 2, and 3. This clique can be written in six different ways: {1, 2, 3}, {1, 3, 2}, {2, 1, 3}, {2, 3, 1}, {3, 1, 2}, and {3, 2, 1}. In general, any clique containing k vertices can be written in k! ways. Examining every clique $k!$ times would have a disastrous effect on the efficiency of any algorithm. Therefore, the algorithm only considers the cliques with the vertices in increasing order. In our example, only {1, 2, 3} is considered. To do this, we build cliques by adding the vertices in order.

Let C be a clique in a graph, and let X be the set of vertices which are adjacent to every vertex in C and which are larger than every vertex in C. We call X the *extension set* of C since we may add any of these vertices to C to form a larger clique. It should be noted that adding two or more elements from the extension set may not yield a clique since these additional nodes may not all be adjacent to one another.

Given the notion of an extension set, we can create a search tree to represent the search space for the maximum clique in a graph. Each node of the tree stores a clique C and extension set X. For each vertex x in the extension set, the node has a child whose clique is $C \cup x$, and whose extension set is the set of nodes in $X \backslash x$ which are adjacent to x and larger than x. The root node of the tree has the empty clique and every vertex is in the extension set.

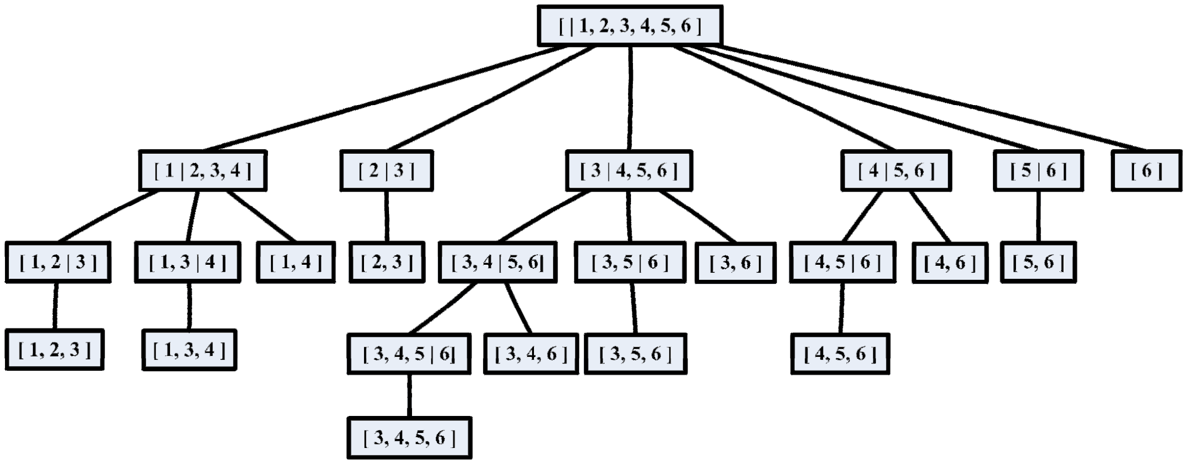

FIGURE 4 The search tree for the graph in Figure 3a. Internal nodes are represented in the format [*C*|*X*], and leaves as [*C*].

A maximum clique in the graph corresponds to a node with largest depth in the tree. We can find a maximum clique by performing an in-order traversal of the tree, keeping track of the largest clique found so far. This algorithm considers every clique in the graph in lexicographical order. However, the maximum clique can be found by traversing the tree in any order we choose. This is the basis of the volunteer computing algorithm. In fact, the algorithm can easily be extended to solve any problem for which the solution is to find a node with a particular property in a search tree. For any problem, one only needs a method to compute the nodes in the search tree and a method to determine whether or not a node has the desired property. Computing the nodes in the search tree can be reduced to computing the children of a given node and identifying the root.

For the maximum clique problem, we have already described how to compute the children of a node, and what the root is. To determine if a node has the property, we simply ask if the given clique (*C*) in the node is larger than that of the current best-known node. For the remainder of the description, we will discuss the problem in more general terms.

Let *N* be a node in a search tree. We say we have *solved N* if we have found a node in the subtree rooted at *N* with the specified property. We can think of solving *N* recursively. To solve *N*, we can solve each child of *N* and pick the best of those solutions, where the definition of best depends on the problem. Notice that obtaining the correct solution does not depend on the order in which the children are solved. Of course, the efficiency of the search might depend on this, and it is important to note that there are also optimization techniques that can (and should) be used to decrease the search space by allowing the pruning of entire subtrees.

The Volunteer Computing Algorithm

The problem with this approach is that these search trees become prohibitively large for even moderately sized problems. By distributing the work to many computers, the time to complete the solution can be dramatically reduced. To do this we must *split* nodes—that is, reduce the problem of solving a node to solving all of its children. Briefly, this works as follows (See Figure 5). A volunteer requests a node to solve. If the node is small enough to be solved completely by the volunteer, the volunteer proceeds to solve it and return the answer to the server. The server then determines if it is the best answer so far. If the node is too large, the volunteer splits the node by giving back to the server the list of child nodes to add to the database (so other volunteers can solve them), keeping one for itself to solve. This process repeats until the problem is small enough for the volunteer to solve. Once all of the split nodes have been solved, the answer to the original problem can be determined. The beauty of this algorithm is its simplicity—we solve all of the children of a node independently and take the best answer. Of course, each child of a node is solved in the same way.

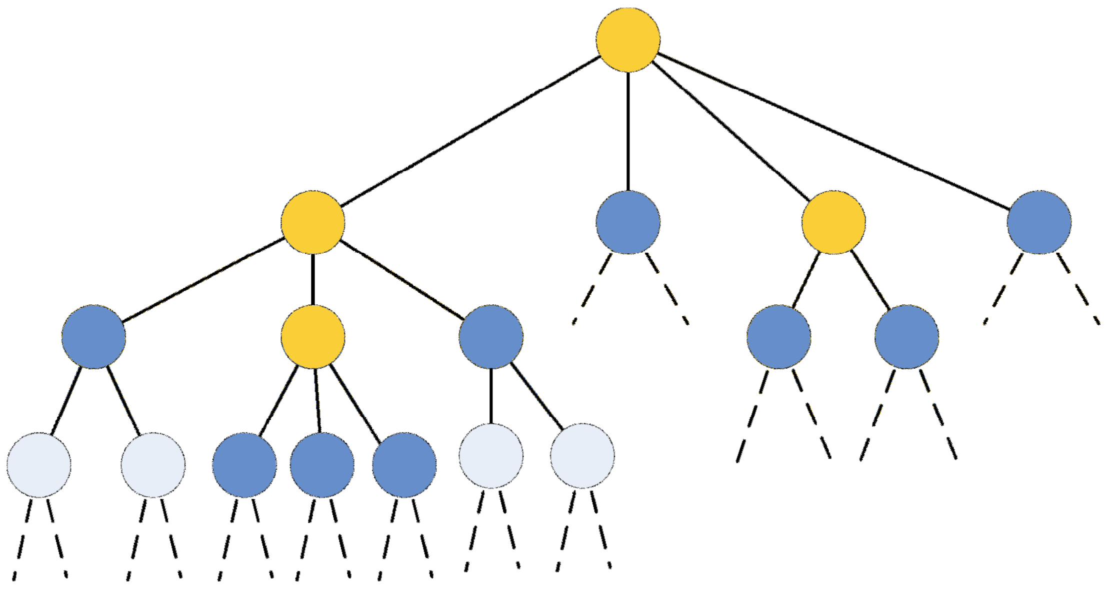

FIGURE 5 A search tree that has been split. The yellow nodes have been split, the darker blue nodes are the roots of subtrees to be solved, the lighter blue nodes are normal nodes, and the dashed lines indicate more nodes below the given node.

There are some important, but perhaps subtle, details about the process of splitting nodes that must be taken into account. We will mention just a few. The way in which nodes are split can have a significant impact on the overall performance of the algorithm. If nodes are split too much, then the problem sizes become smaller, which means the communication overhead becomes more significant, as does the storage requirement of the server. On the other hand, if a volunteer attempts to solve a node that is too large, hours, or even days, of computation might be wasted if the volunteer gives up before finishing. One solution to this problem is to keep track of how long each volunteer spends solving problems of a particular size, and increasing or decreasing the problem sizes appropriately. This is somewhat problematic since the size of a subtree is generally unknown, although there are various methods of estimating it. It might seem like a good idea to attempt to give volunteers smaller nodes to begin with so that too much splitting does not take place. Unfortunately, it turns out that this severely degrades the performance of the algorithm. Once some of the larger nodes are solved, many of these smaller nodes can be pruned.

Although we have not done extensive testing to determine the effectiveness of this volunteer algorithm—particularly in comparison with the parallel algorithm from which it was developed—we feel that it serves the purpose we have in mind here. Our goal is to demonstrate that a casual game can be used to perform volunteer computing, with this algorithm simply serving as our test case. With that in mind, we turn our attention to the volunteer computing game based on this algorithm.

A Volunteer Computer Game

Because the efficiency of the volunteer computing implementation is, above all, dependent on participation, we have created an online, casual game to attract users. In addition to letting their computers automatically work on the problem, players can direct the solving of nodes through their choices in the game.

An Algorithm becomes a Game

We will now describe how we turn the problem of solving a node into a simple game. We decided that it would be important for the user to make “decisions” about which branch of the search tree to explore rather than solving it in a determined order. Therefore, it was necessary to create a game in which the user would make choices between numerous branches in the tree—represented by individual nodes.

In our first implementation, we represented each child of a node by a button. So, if a node has twenty children to choose from, a player is given twenty buttons from which to select. After selecting a node (by clicking a button), the player is given a new set of buttons that corresponds to that new node's children. When the search reaches a leaf in the tree, the algorithm backtracks, giving the player a new list of choices for a previously visited node—with the already explored children omitted, of course. This process repeats until the search tree is exhausted, at which point the answer is returned to the server.

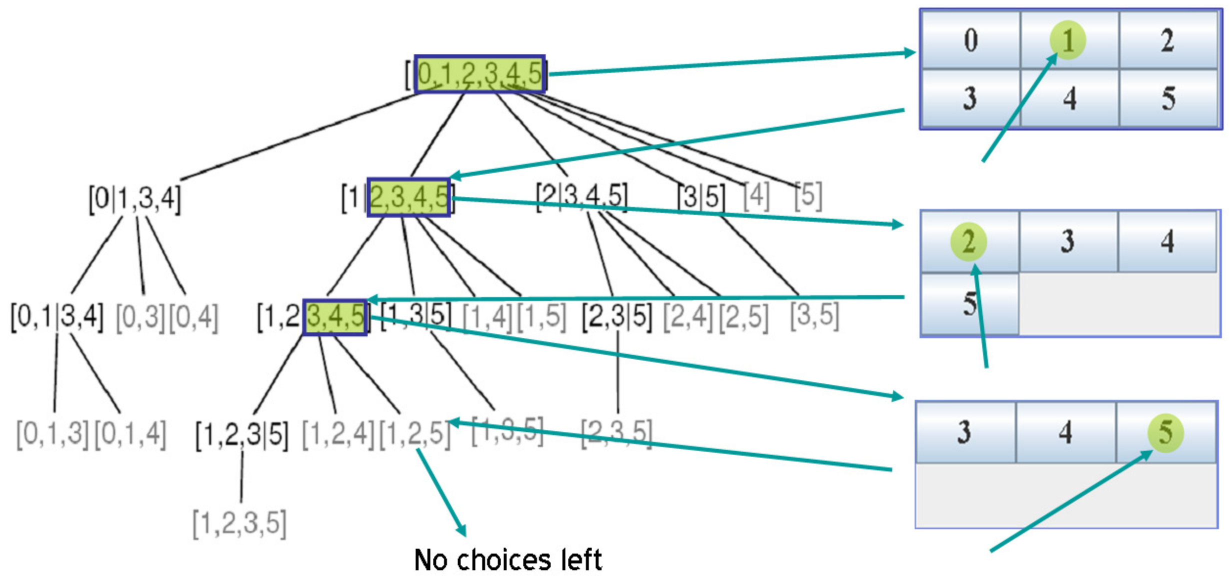

FIGURE 6 How search tree decisions are mapped to buttons.

Of course, *The Button Game* is not very exciting, and not many people would play it for very long. To transform *The Button Game* into a more engaging game, we essentially needed to replace the buttons with more interesting gameplay decisions. In *Wildfire Wally,* a forest is constructed on a grid, giving us the perfect opportunity to make each square in the grid represent a decision in the search tree. Unfortunately, the grid is of fixed size, and the number of children of a node might be much smaller or larger than the number of cells on the grid, making a one-to-

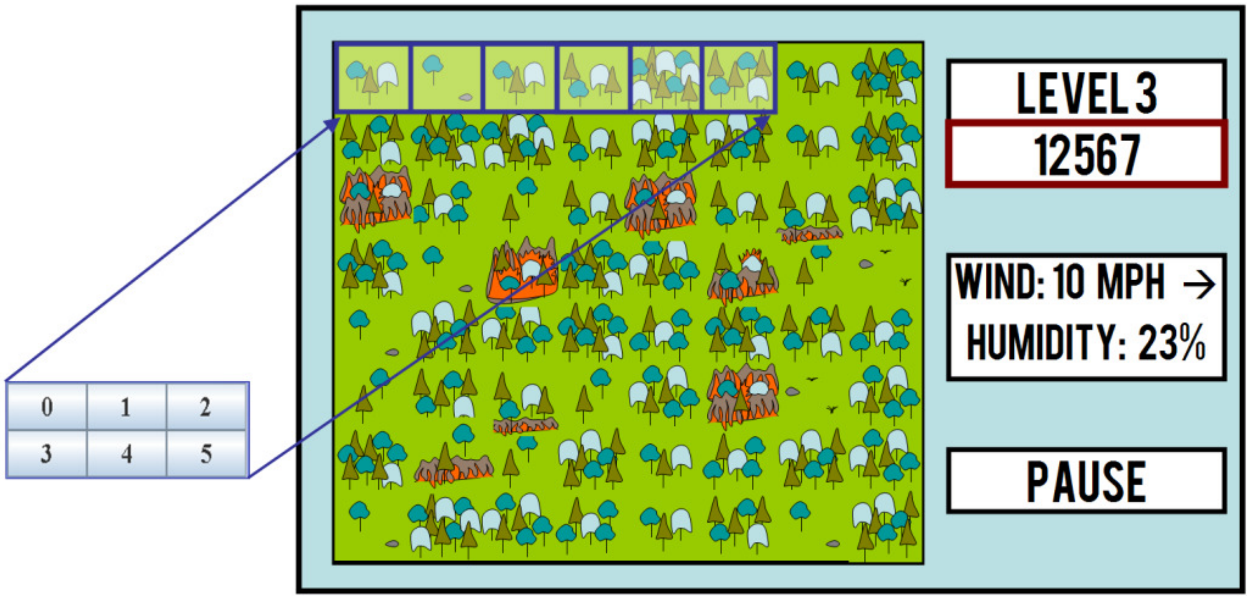

FIGURE 7 Turning the button game into *Wildfire Wally*

one mapping impossible. Instead, we map as many choices to the grid as possible. If there are not enough choices, we simply repeat as many as needed. Thus, before the player clicks on any cell to extinguish a fire, each clump of trees is assigned to a particular node from the current node's children. Then, with the resulting click, a node's child is selected, and the forest is remapped to represent the new node's children. As long as gameplay persists, players make decisions and contribute towards solving the maximum clique problem.

It should be pointed out that players are not really contributing anything meaningful to the computation since they have no way of making intelligent choices. Even if they did, it would be impossible to make choices that would simultaneously be good for both the algorithm and the game. One might argue that we might as well just implement any game, and run the computation in the background independent of the game. Doing so would actually make the computation faster because it would eliminate the time spent waiting for the user. It would also allow us to implement any game we want, with any set of rules, instead of being bound by restrictions dependant upon the particular problem being solved. In a sense, this is absolutely correct. The idea would be that the player is paying (in computation cycles) to play the game. Although this is a great idea that to our knowledge has not been employed, there are advantages to our approach, and to that we turn our attention.

Human versus Computer Contribution

During gameplay, two nodes are actually being solved simultaneously—a small node being solved by the gameplay, and a larger node being solved by the volunteer algorithm running in the background. The most obvious advantage to the background process is that it can solve a node thousands (or even millions) of times faster than the game. In light of the proportionally small contribution gameplay actually makes, it is reasonable to question why we would include human interaction all. There are several motivations behind this decision.

Despite how insignificant human contributions might be in comparison to the contributions of the background process, these actions still accomplish some work in addition to the background process. Even if we discount the work accomplished by the gameplay, the fact that user actions are mapped to algorithmic decisions is important. Although the difference between whether or not the actions of the player contribute to an algorithm might be subtle—and has little to do with gameplay—we think it is critical to the success of the idea. If the actions of the player are contributing in a real way—even if only in a small way—the gameplay becomes more meaningful and players will feel good about playing, which encourages longer playing times. Note that this is an added draw to the game in addition to how enjoyable the game is to play in its own right. Since our goal is to increase participation in volunteer computing, we wish to exploit any feature that will draw and retain players.

The game also gives us another method by which to spur competition and demonstrate progress, since players can be ranked not only by their total contribution to the project through the background process, but also their contribution through gameplay, and how well they score in the game. Finally, while the connection between *Wildfire Wally* and the maximum clique problem is thin, the full power of a volunteer computing game will be realized when it is able to more directly present the problem to users so that they have the opportunity to bring to the game more than their mouse clicks and keystrokes. We expand on both of these ideas below.

Adaptability

During the development of our volunteer computing game, one of the primary design objectives was to create a game that did not explicitly exclude a single gender or age-group, appealing to the widest demographic possible. While we believe that *Wildfire Wally* succeeds in this objective, we also recognize that it is unreasonable to completely ignore gender or gaming preferences. In addition, we wanted the idea to be extendable to as many computational problems as possible. Therefore, our design allows multiple games to solve one problem, and multiple problems to be solved by the same game.

The only communication between the game and the problem solving portion of the application is a list of choices and decisions. Therefore, different games can be plugged in by substituting code in only one place. As long as a player makes decisions in a game, his or her gameplay will work seamlessly with our search tree solving algorithm. Since our foremost goal is to attract as many participants as possible, this feature gives us tremendous flexibility in catering to enthusiasts of different gaming genres. We can create a zombie shooter game that caters to fans of first person shooters, or a relationship based game that is slanted towards those who prefer life-based simulations. As a result, we could theoretically construct an entire gaming website of casual online games that caters to numerous demographics, with their cumulative game actions being used to solve the same problem.

Our design is equally flexible in allowing different problems to map to the same game. Given the amount of time spent developing and producing high-quality online games, it is only reasonable that the valuable contributions of a game extend beyond a single problem. Our design allows us to solve any problem mapped to a search tree by implementing a handful of algorithms for the specific problem we wish to solve. This has a couple of important advantages with regards to volunteer computing. First, if a game is fortunate enough to become wildly popular, it makes sense to utilize this popularity as much as possible. Second, it encourages game design specifically for volunteer computing. If game designers believe their game's shelf-life extends only as long as the problem is unsolved, it would hardly be worth the effort. Mapping different problems to one game increases the game's shelf-life since the game's success hinges on the effectiveness of the game design and is not constrained by the problem itself.

Thus, the main weakness of our game—the fact that the details of the problem are not presented to the player in any meaningful way—turns out to also be one of its strengths. Although we believe that the most successful volunteer computing games will combine the computational power of the user and her computer, we believe that this paradigm also has promise.

Volunteer Computing Game Design Principles

Our framework is only extendable to problems that can be modeled using a search tree, which limits what problems it can be adapted to solve. However, we would like to note that we are presenting this framework merely as one example of using casual games to solve computational problems. Our goal is to spur other researchers on to develop games that can assist in solving a wide array of problems. To this end, it is important to realize that although well-known general design principles apply to any computer or video game, particular decisions become even more critical when designing volunteer computing games. We will use *Wildfire Wally* to help illustrate design decisions that can dictate the success of a volunteer computing game.

Simplicity, Simplicity, Simplicity

Simplicity is a fundamental design objective in any online casual game. However, because it can directly affect three of the barriers of volunteer computing—lack of broad appeal, limited demographic, and lack of technical savvy—simplicity becomes that much more important within the context of a volunteer computing game. We will look at three different dimensions of simplicity: simple to run, simple to learn, and simple to play.

Simple to run translates into making the game as accessible as possible. Because *Wildfire Wally* is a web-based Java applet, anyone who is familiar with basic internet browsing can play. According to Luis Sarmenta (2001), “Java’s applet technology, coupled with its popularity and ubiquity, offers us an opportunity to make volunteer computing both easier and more accessible.” As was previously noted, downloading and installing a game is either beyond the skill set of some users, or is step that some users are unwilling to take. Many people have a media-fueled paranoia of viruses. For that reason, they are reluctant to download any application from a source they do not absolutely trust. Finally, there is a widespread misconception that downloading applications will result in a decrease in computer performance. Since volunteer computing relies so heavily on participation, we want to eliminate as many obstacles as possible that may inhibit involvement.

Simple to learn means that the basic gameplay should be described in one sentence. The goal of *Wildfire Wally* is to "prevent the destruction of the forest by creating fire lines and putting out fires." From this sentence, players can understand the primary goal of gameplay even though the game itself might include additional factors that make it more interesting. While complex games can have a stronger appeal to particular gaming demographics, overall a game will draw from a larger player base if it follows simple mechanics.

Simple to play corresponds most directly to simple game controls. *Wildfire Wally* uses nothing but mouse movement and clicking. If trees are on fire, the player clicks to put them out. If trees are not on fire, the player clicks to cut them down and create a fire line. For someone who is uncomfortable with computer interfaces, a game that relies on rapid keyboard movements or complicated key stroke combinations is not worth playing. Complex controls may appeal to certain users, but it will generally exclude a larger audience than it draws.

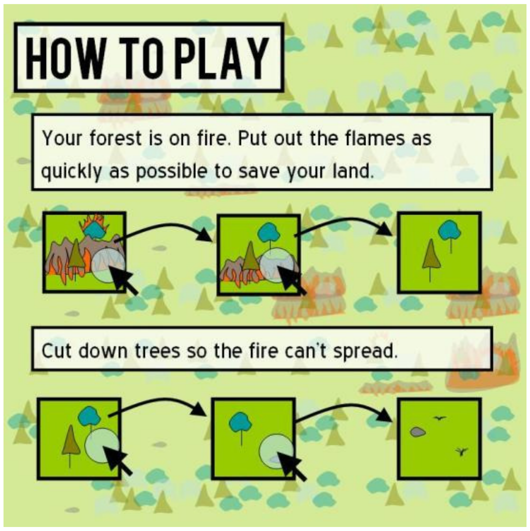
HOW TO PLAY
Your forest is on fire. Put out the flames as quickly as possible to save your land.
Cut down trees so the fire can't spread.

FIGURE 8 The mock-up instruction screen for *Wildfire Wally*

Rapid Gameplay

Since game moves directly correspond to search tree progress, it is reasonable to force the player to make as many game moves as possible as quickly as possible. As gameplay becomes more rapid, the rate at which the problem is being solved increases. In *Wildfire Wally,* players must act quickly to prevent the fire from consuming their forest.

It is important to note that while rapid gameplay is generally good, this should be finely balanced with preventing a high-stress environment. High-stress, fast-paced games appeal to a relatively small target audience. We try to maintain this balance in *Wildfire Wally* by keeping levels fairly short, allowing the player to catch their breath between spurts of play.

Showing Progress

According to Salen and Zimmerman (2004), “progress is the rhetoric of western culture.” Consequently, it is important to demonstrate to players that game actions are not trivial—that there is meaning behind their actions and are rewards for their struggles. In most games, this is displayed as a numeric score or visual representation of their accomplishment.

In volunteer computing games, progress should be shown in two different ways. First, progress in the game itself should be made known. However, players should also be aware of their contribution towards the larger problem. *Wildfire Wally* utilizes both of these indicators. One score corresponds to how successful they have been in protecting their forest in each successive level. The second score corresponds to the total contribution in solving the maximum clique problem. We could further refine this second score by showing their contributions to the problem through gameplay and the background process separately. Arguments could be made in favor of both methods.

By demonstrating that actions have implications not only on a personal level, but also on a far greater scale, players will find more profound meaning in their gameplay and begin to gather a sense of ownership in the problem. This keeps users playing the game and contributing to the problem even if the novelty of the game has worn off.

Replayability

Since large problems will take a considerable amount of time to solve regardless of the number of people contributing, a central goal of game design should not only be to include as many players as possible, but to keep these players coming back as often as possible.

Gameplay cannot be entirely static – that is, the game should not remain exactly the same from one play to another. There must be enough variation to keep players engaged over long periods of time. We try to accomplish this in *Wildfire Wally* in three different ways.

First, we introduce variables that distinguish gameplay from level to level, and game to game. There is a humidity factor that either increases or decreases the likelihood of neighboring trees catching on fire. There is also a wind factor which can have acute effects on the direction a fire is spreading. These variables are altered from level to level, requiring the player to adjust accordingly.

Second, we introduce a random element into gameplay. Every time a player plays *Wildfire Wally,* they receive a randomly configured forest, with the fire starting at a random location in the forest. This forces players to conceive new strategies for protecting their trees, and prevents players from concocting a dominant strategy to win every time. In addition, many of our game rules are based on likelihoods instead of absolutes. So instead of writing game rules like:

IF A is on Fire AND B is next to A THEN B *will always* catch on fire

We write game rules like:

IF A is on Fire AND B is next to A THEN B *has a 30% chance* of catching on fire

Even if a user receives the same forest configuration with a fire starting in the exact same spot, there will still be variation in the way the fire spreads.

Third, we designed the game to be easily extendible. Introducing new variables in the game or objectives to the players (perhaps as levels increase) is fairly straightforward. Currently the only objective in *Wildfire Wally* is to put out the fire and save the forest. In future versions, we will add elements like rain that can help extinguish the fire, thunderstorms that might spark new fires, campgrounds with people who might accidentally start a fire, a fire tower that decreases fire likelihood within a certain radius, houses that must be saved, and geological barriers such as rocks, rivers, mountains, and grasslands. Each of these possible features would force the user to determine new strategies, extending the length of interesting gameplay.

It is important to note that none of these features drastically impact the simplicity of the game. To win, the player must still "prevent the destruction of the forest by creating fire lines and putting out fires." We are merely adding slight twists and variations to maintain dynamic gameplay.

Competition

Some of the most successful game titles use direct competition. Even in simple games, people become more engaged when competition is involved. A high scores list, as mentioned, is one possibility, while head-to-head matches and team competitions are others. Just as showing progress is important in both the game and the problem, competition can be based on gameplay, contribution to the problem, or a combination of both. Special recognition can be given to players (or teams) who pass a certain contribution level first or who have played for the most consecutive days. Some players will continually engage themselves in a game even if they do not find it particularly entertaining, just for the sake of competition. Finding ways of creating competition, especially ways in which every user has a chance to be recognized, is critical to the continued success of a volunteer computing game.

Community

Between social sites such as MySpace and Facebook and email, the internet thrives with social interaction. Gaming is no different, and opportunities exist for creating a sense of community within the volunteer computing games culture. Allow players to combine their efforts mid-game (maybe even rewarding and ranking teams for their cumulative effort towards the problem). Create message boards or wikis to discuss the game and the underlying research. Keep players informed of new features of the game, etc. through opt-in e-mails. Even if players do not have an immediate interest in playing the game, they will participate more frequently just to remain involved in the community.

Future Work

Human Insight

In volunteer computing, the main goal is to harness the computational power of the user's computer. In human computing, the goal is to harness the computational power of the user. With volunteer computing games, ideally the goal is to use games to attract users to assist with important research so that the computational power of both users and their computers can be harnessed. Although volunteer computing games can be useful without utilizing human insight, we believe that the full potential is realized with the synergy created by combining games, brains, and silicon. We term these games *human computing games*.

In *Wildfire Wally,* gameplay works toward solving the maximum clique problem. However, players have no visual knowledge of the problem. While their decisions have meaning within the context of the game, their gameplay cannot solve the maximum clique problem in an intelligent manner. Because of this, *Wildfire Wally* serves only as an elementary example of the power of a volunteer computing game. It may be possible to create a game based on search trees that can utilize human insight. For instance, the ordering of the vertices in our algorithm is arbitrary, but if we allow the player to choose the order, it may mean that more subtrees can be pruned quickly. In addition, if the player is able to make intelligent choices about which branch to take, they may find a better solution much quicker than a random or in-order algorithm. These are some of the things that need to be explored further.

We are in the beginning stages of designing a game to solve a graph pebbling problem that presents the problem directly to the player, allowing a brilliant player to discover a more efficient way to complete the problem than the existing solution. In this case one of our goals, like that of the *Fold It!* team, will be to study the strategies of players so we can develop better algorithms.

In both *Wildfire Wally* and von Ahn's games, all players can contribute to the solution of the problem. A well designed volunteer computing game would also allow an exceptional player to contribute even more, potentially helping develop a new algorithm. Presenting a problem to a large number of people who would otherwise never see it means there is much more potential for someone to see the problem differently. Because many of the players will have had no formal training related to the problem at hand, they will have no preconceived notion of what may or may not work. This may actually be advantageous—they will try strategies that those "in the know" would never attempt for one reason or another.

The major difficulty with this approach is that presenting a computationally complex problem is not generally easy. Humans can easily recognize objects in two-dimensional images, but can they find a maximum clique in a large graph? How does a designer show the player a graph on 5,000 vertices? How does he present to them a 200 digit number so that they can attempt to factor it? These are the sorts of questions that must be addressed in order for volunteer computing games to realize their full potential.

Massively Multiplayer Online Games

Our research has focused primarily on using casual games in volunteer computing. However, Massively Multiplayer Online Gaming (MMOG) also provides a wonderful opportunity for volunteer computing. Thousands of users simultaneously participate in games such as *Everquest* and *World of Warcraft*. In fact, as of March 2004, current *Everquest* players had already amassed a total playing time of 184,000 years (GameZone, 2004). In addition to their tremendous popularity, MMOGs have several strengths that cater specifically to volunteer computing.

MMOGs already present worlds of substantial size and complexity. This accommodates the size and complexity of larger instances of problems like maximum clique, allowing game designers to present problems in increasingly creative and innovative ways. For instance, the nodes of a graph can be represented by locations in the world (kiosks, computer stations, NPCs, etc.), with the edges being represented by paths or network connections between these locations.

Nearly all MMOGs contain well established social groups. Players unite in *guilds* (long term) or join *parties* (short term) for social interaction, or to complete difficult tasks that cannot be tackled individually. Many guilds enjoy competing for power within the construct of the game. This means that large groups of players are already spending large quantities of time trying to solve complex problems. These factors form an ideal environment in which to present volunteer computing problems. A mechanism can be created to allow players to interact in a cooperative algorithm to solve a large problem. We have a few ideas about how we might present graph problems, for instance, in a MMOG so that players can work together to solve them. In addition, when a group of users form a party, not only can they work together to solve a problem, but it may be possible to utilize their computers much like a cluster, at least for a brief period of time.

Finally, MMOGs encourage *emergence*. Game emergence is the unexpected behavior or strategy that arises out of gameplay, or a player adaptation in the game that developers may not have considered. Players are incredibly resourceful in how they interact with their environment, especially in light of competition and personal progress. Our hope is that through gameplay, players will discover more efficient ways to solve the problems given to them—that they will come up with strategies that we could not envision.

Conclusion

Demonstrating problems in relevant and engaging ways is not new. Mark Twain's Tom Sawyer presents the job of painting a fence in such an appealing way that the neighborhood boys pay Tom for the "opportunity" to whitewash the fence. Similarly, we believe that the world of online casual gaming presents a remarkable opportunity for volunteer computing. The nearly universal appeal of casual gaming can be exploited to increase the computational power available to researchers by bringing an entirely new demographic group into the fold. In addition, well designed games can introduce to the players the concepts behind the research in a non-threatening way, which may entice them to get even more involved. Finally, the computational power of the players themselves has the potential of making dramatic contributions to many areas of research. There is much work to be done before the full power of volunteer computing games will be realized, but the ground has definitely been broken.

Acknowledgements

This research was supported by the National Science Foundation grant CNS-0353566. We would like to thank Chris Martens and Priyanshu Mutreja for their contributions to the initial prototype of *The Button Game*.

References

Anderson, D. P. (2003). Public computing: Reconnecting people to science. *Proceedings of the Conference on Shared Knowledge and the Web.* Residencia de Estudiantes, Madrid, Spain.

Anderson, D. P. (2004). BOINC: A system for public-resource computing and storage. Proceedings from *Fifth IEEE/ACM International Workship on Grid Computing.* Pittsburgh, PA, 4-10.

Anderson, D. & McLeod, J. (2007). Local Scheduling for Volunteer Computing. *2007 IEEE International Parallel and Distributed Processing Symposium.*

Androvich, M. (2007, July 18). *Nickelodeon to invest $100 million in casual gaming*. Retrieved July 21, 2007 from http://www.gamesindustry.biz/content_page.php?aid=26856.

BOINC. (n.d.). *Survey results*. Retrieved July 18, 2007 from *http://boinc.berkeley.edu/poll_results.php.*

Cusack, C., Martens, C., & Mutreja, P. (2006, October). *Volunteer Computing Using Casual Games*. Paper presented at Future Play 2006 International Conference on the Future of Game Design and Technology, London, Ontario, Canada.

Entertainment Software Association. (2006). *2006 Sales, Demographic, and Usage Data: Essential Facts About the Computer and Video Game Industry.*

GameZone. (2004, March). *Everquest Gears Up for 5th Anniversary Celebration*. Retrieved June 25, 2007 from http://pc.gamezone.com/news/03_15_04_02_07PM.htm.

Internet World Stats. (2007). *World Internet Usage and Population Statistics*. Retrieved June 30, 2007 from http://www.internetworldstats.com/stats.htm.

Justiniano, C. (2005, October). Tapping the Matrix: Revisited, *Presetned at the BoF Linux Forum,* Copenhagen.

Law, E., von Ahn, L., Dannenberg, R., & Crawford, M. (2007, September). TagATune: a Game for Sound and Music Annotation. *Proceedings of ISMIR 2007: International Conferences on Music Information Retrieval and Related Activities*. Vienna, Austria: OCG.

Macrovision. (June 28, 2006). *Survey Reveals Casual Gamers are Not So Casual*. Retrieved July 3, 2007 from http://www.macrovision.com/company/newscenter/pressreleases/1434_pr2006_22.htm.

Marketing Charts. (May 14, 2007). *Study: Brands Must Adapt to Shifting Media Habits of Users*. Retrieved June 28, 2007 from http://www.marketingcharts.com/topics/blogs/study-brands-must-adapt-to-shifting-media-habits-of-users-369/.

Salen, K. & Zimmerman E., (2004). *Rules of Play, Game Design Fundamentals*. Cambridge, Massachusetts: The MIT Press.

Sarmenta, L., (2001). *Volunteer Computing*. Ph.D. Dissertation, Massachusetts Institute of Technology.

Thimm, L., Kreher, D., and Merkey, P. (2006). A Parallel Implementation for the Maximum Clique Problem. *Journal of Combinatorial Mathematics and Combinatorial Computing, 63,* 183-207.

Thompson, C. (June 25, 2007). For Certain Tasks, the Cortex Still Beats the CPU. *Wired Magazine*, Issue 15.07.

von Ahn, L. & Dabbish, L. (2004). Labeling images with a computer game. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 319-326), New York, NY: ACM Press.

von Ahn, L. (2006, June). Games with a Purpose. *Computer, 39*(6), 96-98.

von Ahn, L., Ginosar, S., Kedia, M., Liu, R., & M. Blum, M. (2006). Improving Accessibility of the Web with a Computer Game. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems* (pp. 79-82). Montreal, Quebec, Canada:ACM Press.

von Ahn, L., Liu, R. & Blum, M. (2006). Peekaboom: A Game for Locating Objects in Images. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in Computing Systems (pp. 55-64).* Montreal, Quebec, Canada: ACM Press.

von Ahn, L., Kedia, M., & Blum,M. (2006). Verbosity: a game for collecting common-sense facts. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems (pp. 75-78).* Montreal, Quebec, Canada: ACM Press.

von Ahn, L., Ginosar, S., Kedia, M., Liu, R., & Blum, M. (2007, April). Improving Image Search with Phetch. In *ICASSP 2007: IEEE International Conference on Acoustics, Speech, and Signal Processing, 4 (pp. 1209-1212). Honoluli, HI.*

Wallace, M. & Robbins, B. (Eds.). (2006). *2006 Casual Games White Paper*. IDGA.